Algorithmique II
Année 2006-2007

Florent Bouchez
Jean-François Pineau

TD n°2

1 Échauffement : compter en base d'interrupteurs

On considère une ligne de n interrupteurs tous en position fermée. On s'amuse ensuite à compter en base 2 avec ces interrupteurs, en partant de 0, jusqu'à $2^n - 1$. Calculez le nombre total de changements d'état des interrupteurs, au choix :

- en comptant le nombre de changements d'état quand on passe de m à $m + 1$,
  Ça va pas la tête ?
- en sommant le nombre de changements d'état de chaque interrupteur,
  Interrupteur i change toutes les $\frac{1}{2^{i-1}}$ fois
- par de l'analyse amortie.
  Placer 2 euros pour ouvrir un interrupteur. Un pour l'ouvrir, l'autre pour le refermer plus tard. Donc 2 euros utiliés à chaque passage de m à $m + 1$.

Au total, $2^{n+1} - 2 - n$ changements.

2 Plat principal : recherche de mots

Dans cette partie, on se donne un ensemble de vrais mots de la vraie vie, D. Par exemple "tas", "tige", "table", "tdman", "tabouret"... L'alphabet que nous appelerons Σ est alors celui que vous connaissez depuis que vous êtes petits, avec toutes les lettres de 'a' à 'z'.

Question 1 Quel est le cardinal de Σ ?[1]

26 □

On cherche un moyen de savoir si un mot m appartient à D, de préférence rapide. On veut aussi, si $m \notin D$, trouver u préfixe de m de taille maximale et tel qu'il existe $v \in \Sigma^\star$ tel que $uv \in D$.

[1] Attention, il n'y a pas de piège.

Question 2 Réecrire cette question en langage compréhensible et donner un exemple qui clarifie ce charabia. Pouvez-vous utiliser le TD n°1 pour résoudre ce problème ?

Cela correspond à la recherche d'un mot dans un vrai dictionnaire, si le mot existe, on le trouve, sinon, on reste bloqué entre les deux mots qui lui ressemblent le plus et on peut dire à partir de quelle lettre le mot est faux. Pour la partie existence, on pourrait utiliser une table de hachage, mais le caractère non ordonné des tables de hachages ne permet pas de trouver facilement où est la première erreur dans le mot. □

Question 3 Est-ce que ça vous fait penser à un automate ? Si oui, représentez cet automate sous forme d'arbre. Si non, retournez à la question **??**.

Par exemple pour "tas" et "table", trois états à la suite avec des transitions étiquettées par "t" et "a", puis bifurcation au niveau "s" et "b" vers deux branches différentes. □

Question 4 Pensez à une implantation d'une structure de données pour votre arbre.

Un type nœud qui contient une liste d'association (lettre, nœud suivant). Complexité $O(|m| \times |\Sigma|)$. □

Question 5 Vérifiez que vous n'arrivez pas à avoir une complexité en $O(|m|)$ pour la recherche d'un mot m. Souvenez-vous maintenant que vous n'avez pas de contrainte de taille, et donc vous pouvez utiliser toute la mémoire que vous voulez. Trouvez donc une meilleure structure qui vous permette d'atteindre cette complexité.

Tableau de plein de lignes de taille chacune $|\Sigma|$. *Chaque ligne est un nœud de l'arbre, chaque case pointe vers* `null` *si pas de branche correspondante, sinon vers l'adresse de la ligne correspondante. Pour* M *mots dans le dictionnaire, tableau de taille* $O(M \times |\Sigma|)$. □

Question 6 Rendez-vous compte avec horreur du temps que prend l'initialisation de votre structure de donnée. Inventez un moyen qui vous permette de n'initialiser que les données qui vous intéressent.

Pour cette dernière question, pas d'indications écrites sur papier mais à l'oral, la progression des idées doit être la suivante :

– *marquer si les liens sont valides ou pas*
– *la marque doit se trouver à l'extérieur du gros tableau*
– *l'intérieur doit pointer vers la marque*
– *la marque aussi doit pointer vers l'intérieur*

Ce qui permet de savoir quelles cases du tableau sont initialisées en vérifiant si dans une case, le pointeur va bien vers la pile a l'extérieur, et que dans cette pile la case re-pointe bien vers la case originelle du tableau. □